電腦哲學家
—— 符號邏輯與人工智慧的探討及實作

國中組應用科學科第一名

台北市北政國民中學

作　　者：唐宗漢
指導教師：劉更生

一、研究動機

1993年7月間（作者小學六年級時），閱及「有趣的人工智慧程式」中之FOL（註1）程式FETCH，深覺其功能之侷限與不足；因此有人工智慧及邏輯方面之興趣，故期能加以改進，做出具完備符號邏輯推論能力之程式。

惜當時程式設計能力未逮，故僅撰一程式架構（LOGIC），以處理三段論法規則（Syllogisticorules）之一種（註2），經一年半之改進，並數度改寫其「推論機制」後，終於做出具「推論智慧」、接受多語混合輸入的「三段論法處理界面」［SAPI］與「邏輯推論界面編譯器」［LPIC］（註3），並能處理多種不同邏輯關係之程式後，始達成原先期望之目標。

［SPAI］即成，即以此報名參加校內科展。但因所學有限，致難有長足進步，常以爲憾。及閱國外此類軟體之較新（1994）文獻，恍然大悟之餘，遂將［SAPI］之架構與功能擴充；因［SAPI］架構極「硬」（缺少彈性、擴充性），故未能改寫之處，撰一名爲［IPAS］（註4）之獨立軟性架構程式以實作之；除取截長補短之效外，更以電腦實作維根斯坦＜邏輯哲學論＞中之邏輯世界爲期。

二、研究目的

1.實作以符號邏輯規則推理之程式，使其以「符號邏輯演算法」處理符號間之關係。
2.使程式具接受「自然語言」敘述之能力，並探討處理多國語言之方法。
3.使程式朝「多功能邏輯界面」之方向發展，並探討其擴充之能力與彈性。
4.藉此加深對維根斯坦早期學說之瞭解，及此說程式實作之可能。
5.探討並比較兩類處理符號邏輯推理之方法（「硬式」與「軟式」），並評比目前此類程式之優劣、探討今後改進之方向。

三、使用器材

1.Intel 80486 Sx-25(Notebook), QBasic v1.0, DOS version 5.00, Hercules。

2.Intel 80486 Sx-33, QuickBasic v4.5(C, E), DOS version 5.00, VGA, 倚天中文。

3.Intel 80486 Dx2-25, QuickBasic v4.5(E), DOS version 6.20, SVGA, 倚天中文。

（VisualBasic v2.0, Windows version 3.10）

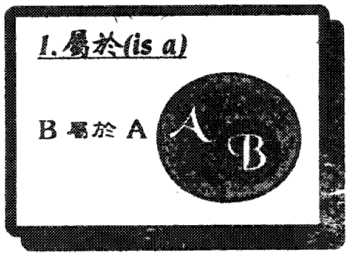
四、研究方法

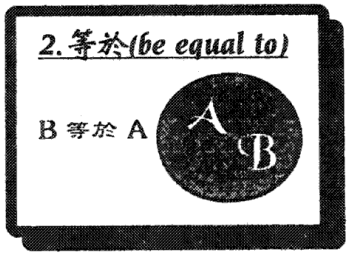
1.SAPI－演算法與架構

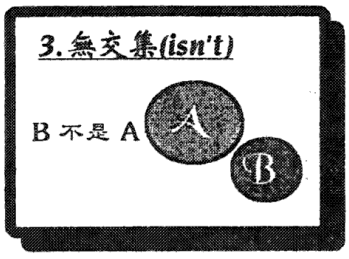
Syllogistic Arguments Process' Interface（SAPI），顧名思義，是以處理三段論式（Syllogistic rules）爲主的程式界面。SAPI的特點有下列幾項：

A.客體、集合間之多樣關係

在SAPI中，客體名與集合名是同等的；也就是說，（單獨的）客體被視作只含一個元素的集合，與其他集合統稱「物」（Object）。物的屬性（如顏色）亦以函屬關係表示，如「紅蘋果」與「紅臘燭」皆屬於「紅的（客體）」此一集合等。SAPI中共有三種事物間的關聯：





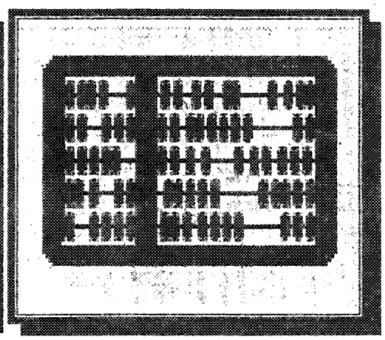


本程式採取的儲存方式爲定義三個陣列LogicOF$（n,2），LogicEQUAL$（n,2）及LogicNOT$（n,2）以存於「子集合」與「母集合」（當然，在推論時有（A==B）<=>（B==A）及（A!=B）<=>（B!=A）之名。這種方法考慮了兩物間所有可能的（集合論上的）關係（註5），使得假言三段論式擴充爲下列11種型式：

名稱[注1]	規則	名稱	規則	名稱	規則
Babara	a => b-b => c, a=>c	*Darii*	a => b-b != c, a != c	*Mazurka**	a => b-b == c, a=>c
*Sinyis**	a != b-c => b, a != c	*Diuling**	a != b-b == c, a != c	*Celarent*	a -> b-b => c, a->c
Ferio	a -> b-b != c, a != c	*Endued**	a -> b-b == c, a->c	*Guava**	a == b-b => c, a=>c
*Murii**	a == b-b !=c , a != c	*Suede**	a == b-b ->c,a -> c		
[注1] 名稱中母音的意義: 'a'-is 'i'-isn't 'e'-some is 'u'-all 'o'-some isn't 'y'-inverse_is					
[注2] 打*號的爲SAPI中有別於三段論法規則而自創的詞。					

而SAPI則以「遞迴查詢」的方法處理以上全部的規則；例如處理is a關係的副程式如下圖，十分簡便、直觀且迅速（假設僅有is a此種關係時）：

```
DEFINT A-Z
SUB CompareOF (Item1$, Item2$)
  SHARED Flag              'Flag 爲命題之眞假值, 0 爲假, -1 爲眞
  FOR LoopCount = 1 TO Towhere  'Towhere是LogicOF$()中所有前提之個數
    IF Flag = -1 THEN
      EXIT SUB             '若命題已知爲眞, 則退出
    END IF
    IF LogicOF$(LoopCount, 1) = Item1$ AND LogicOF$(LoopCount, 2) = Item2$ THEN
      Flag = -1
      EXIT SUB             '若命題判斷爲眞, 則退出
    ELSEIF OF$(LoopCount, 1) = Item2 THEN   '如果集合屬於 Item2$, 則
      CompareOF Item1$, OF$(LoopCount, 2)   '判斷 Item1$是否屬於此集合
    END IF
  NEXT
END SUB
```



B.自然語言處理與多國語文

SAPI採用一種我稱爲「強制語法轉換」的自創方式來處理多國自然語言。「強制語法轉換」的運作方式略如下圖（隨著句型的不同，也有不同的詞語代換方式）：

SAPI V2.60（尚未完成）處理多國語文訊息的方式是將各種句式（sentence pattern）儲存在一個名爲SPATTERN.SAP的檔案中，在使用者輸入句式時逐項檢驗並判別所用語文（中文的判別方式則是先檢查有無ASCII 127+的字元；因SAPI會將英文、德文全轉成大寫，但在處理中文就會遇上麻煩），而使用不同的語文回應。「強制語法轉換」的好處是：只需要設定句式及代換的詞語，只需一個副程式就可分析多國文字的語句

，較寫許多套「SAPI中文版」「SAPI英文版」「SAPI安拉地度拉達皮文版」等，均節省不少精力與效率…

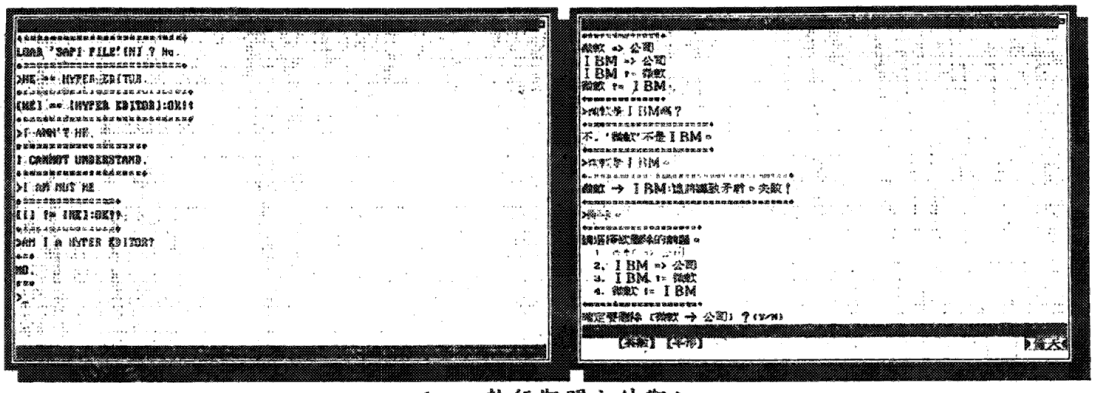
此外，在設計SAPI時我十分注意語句的一些變化，如“Tom and John are both students”其實應該拆成兩個句子；而如「複數形」等也是必須考慮的（詳見附件三）

C.使用者界面

SAPI的使用者界面傾向於「交談式環境」（dialog environment），由使用者輸入語句，程式則根據句子型式的不同，而作出不同的反應。句子共有下列幾種型式：

- 敘述句：即SPATTERN中規定的句式，SAPI會將之轉換並儲存於記憶中。
- 命令句：SAPI提供不少功能以方便使用者操作（見附件二），和敘述句都必須在其後加“.”以與詢問句區別，SAPI提供SimDOS（OS shell）、DIR、SAVE/LOAD等功能及十分完善的線上多語輔助說明。
- 詢問句：SAPI的詢問句式共有三種：一般詢問句（如“Am I crazy?”，“Is Mozart a composer?”）、查詢句式（“Request Tom.”、“Who is he?”）及「查詢子集」（Request Instance）句式等。

SAPI的提示符號爲“>”，使用者在其後輸入語句，SAPI的回答則放在兩個「框框」中（見圖）；IPAS則使用和SAPI幾乎完全相同的操作環境。



(SAPI執行期間之外觀)

2.IPAS－演算法與架構

IPAS（Immediately Process Applying Symbols）與SAPI不同之處，在於其架構偏於「軟性」，且規則及句式皆由使用者自行定義，與SAPI之

「預設句式」頗有不同。

IPAS約有下列數項特點：

A.語句之輸入與分析

IPAS內部有十個保留字（reserved words, 皆以“～”開頭），除此之外，對於使用者輸入的語句無甚「成見」，一切皆由使用者「塑造」而成。

當IPAS接收到使用鍵入的語句時，會先比對目前已有的句式，若未找到符合之句式，則會要求使用者輸入句式（當然，使用者也可將此句當成「宣告式」，整句當作一個客體），並整句連同句式號碼一同儲存；再來是分析句中所包含的客體（「原子事實」），並將其存入貯放客體名稱的陣列中；一個很重要的原則是絕不能有兩個作爲事實的命題等價、矛盾，或互爲 —— 因果這也是維根斯坦所提及的：從一個基本命題不能推論出任何其他的基本命題。（註7）

使用者可以藉“～var”、“～def”及直接輸入基本命題（註8）來定義單詞（及複合詞）之詞性，或在輸入命題時由程式自動判別亦可（詳見下文）。

B.推論規則之建立與推論方法

首先，當程式－不論什麼原因－要判斷命題的真假時，它會先查看有無與此命題完全相同或互爲矛盾的前提，若有則立刻得出答案、結束搜尋。若無（通常如此）則搜尋記憶中的「推論規則」，據此判斷命題之真假。

此即爲此版之創新處。上一版本之作法爲另建一陣列，僅儲存由使用者輸入之「若－則」關係；推論時則將所有的客體代入「結論」部分的變數並查看是否與欲查詢之命題相符，若相符則查詢「前提」部份之真值，若爲真則結論當然亦真，若無法判定則繼續搜尋－此亦爲Tutor及SIR的作法（SIR不能配合真值函數運用）。

此法之缺點在於其處理範圍僅限於呆板之IF…Then關係，如由「A or B」及「～A」導出「B」之程序即超出其處理能力。所以v1.30之作法即有所不同。

v1.30將廢除If…then句式，直接將變數及真值函式應用在敘述句中（真值函式以上述之「圖式法」定義），推論方法則是「試誤法」：設欲查詢之命題設爲真，若代換結果發現與前提之一（或許多）相悖，則此命題爲假；反之，設欲查詢之命題設爲假，若代換結果發現與前提之一（或許

多）相悖，則此命題爲真；兩者皆不相悖，代表其真值未定，如此便能運作並驗證所有邏輯推理之規則（如「否逆定律」等）。

C.句式之建立與鑑別

IPAS之一大突破即爲「句式」方面之處理。早期（v1.20之前）之IPAS將所有語詞一律視爲「客體」，於代換時「萬法平等」，皆可置入變項，導致程式於處理“A IS IS IS”此類文法錯誤（syntax error）之句時發生無窮遞迴（怎麼可能證明如「潮董昱勒睫碣火」之類的無意義命題－嚴格說來是「僞命題」－呢？），終致Stack overflow，天怒人怨。

v1.22迄今，我採用一種稱爲「規定文法」的策略；每個單詞都有其詞性（可以叫noun、verb、adjective，也可以稱作主詞、副詞、受詞），也可以有不只一個的詞性（如“can”、“being”等字）。使用者輸入語句後，先於記憶庫中搜尋有無相符之「句式」（如“noun verb adj noun”或“adj noun ～IS？ noun2”（註9））；若有（可能不只一個），則讓使用者選擇匹配之句式，或自行鍵入句式（若無相匹配之句式時亦同），由程式自行判斷各詞之屬性，如令“COLONIZATION IS BETTER THAN CIVILIZATION”與句式“n b ac2 n”相匹配時，IPAS將自動賦予“COLONIZATION”及“CIVILIZATION”「n」的屬性，“IS”則爲「b」，“BETTER THAN”則視爲一個單詞，具“ac”的屬性等。

如此之好處在於：輸入句子及判斷真值時僅處理「合法」之語句，若再配上v1.30之「辯證求值法」（取其「正反合」之意），則幾無stack overflow（遞迴之多於三層者，幾希！）之狀況發生；且亦可避免使用者「不小心」輸入不合文法卻無查覺之命題，故堪稱一有效之獨創解法。

3.問題之發生與解決

以下是我在研究過程中曾經發生的問題，及採取之解決方式：

A.SAPI部份

- 「查詢」功能之實作（93/07/17）

「查詢」功能是SAPI眾多指令中最早出現的一個，最初僅供查詢“OF”關係，其餘爲日後逐漸增加。解決的方法爲：先搜尋所有與「欲查詢物」有關係之前題，再將查詢到之客體當作參數，再呼叫一次查詢函式（即「遞迴」技巧）即可。

- 「刪除」功能之實作（93/07/19）

0.12版前SAPI的前提儲存在固定大小（100*2）的陣列中，每次找尋

都得For Lm=1 to 100（即使只存了一個前提），遇到“ ”即停止；但爲提供「刪除」功能，故增加變數Towhere以鑑別目前的存量「到哪裡」，故「刪除」時僅需將Towhere-1，而後把每一個在「欲刪命題」後的前題都往前「搬」一個位置即可。

• 「無窮遞迴」之防止（94 / 03 / 29）

「遞迴」功能有個危險：若是副程式重複兩次Call自己，則遞迴過程將會無休無止地持續，直到發生“ Stack overflow ”（堆疊溢位）爲止。SAPI及IPAS的解決方式爲：在Call時將此次參數加至一個字串中，每次要Call自己時先檢查參數是否用過，若是則不再重試。

• 「SOME」關係之加入（94 / 10 / 05）

初擬加入SOME關係時，本欲另建Somels$0等三陣列以貯放，但推論時極易發生問題，故定義“ SOME xxx ”爲客體“ xxx ”的特殊型式（special kind），在貯放及演算時視同「一般型式」處理，而輸出時則以不同方式進行。

• 「Program-memory overflow」之防止（94 / 10 / 21）

SAPI 1.72版時，BASIC的編譯器（BC.EXE）因程式太大而拒絕編譯（Program memory overflow）；經檢查後發現「輔助說明」及「顏色定義」等字串都Include在程式中，佔用極大空間，故採用「訊息外掛」方式，將所有訊息建爲檔案，由程式中讀取。時至今日，SAPI已容許多語輔助說明分別外掛，及使用者對操作界面自行定義等等，殆爲此舉之功。

餘如「中文系統之特殊處理」、「存 / 讀檔機制」等因僅爲小技，篇幅所限，暫不贅述。

B.IPAS部份

• 「變數」之代換（95 / 02 / 03）

IPAS的特點之一在於其可以處理無窮多個變數；但在處理「變數代換」時卻無法事先決定用到了多少變數；偏偏QB又不提供這樣的語法（見右圖）：

```
for Lm=1 to Sum
   for Ln(Lm) =1 to ITowhere
next
....
for Lm=1 to Sum
   next Ln(Lm)
next
```

至於解決方式則十分有趣：因爲共要執行ITowhere^{sum}迴圈，故直接設一個由1至ITowhere^{sum}的迴圈即可；每一次代入的變數序號則

以Lm在Sum進位的位數爲準（如：有五個變數時則以1～（總客體數）[5]的5進位表示法的每一位數，代入相應的Item$序號－“02133”表示第一個變數代換成序號0的客體，第二個代換爲序號2的客體，餘類推）即可。

- 「堆疊溢位」之防止（95/02/07）

與SAPI一樣，IPAS在實作時也會遇到Stack overflow的問題，且比SAPI更嚴重。IPAS的IsTrue函式當遇到真值函數時，便要重複呼叫自己，以確定作運算的兩個命題之真值，極易造成堆疊溢位；直至「句式」架構建立完畢後，方不再爲此所擾。

我被這個問題困擾了三天（當時的IPAS只能處理「三個客體」，不能再多），最後除了儲存「曾經處理過的參數」（SAPI如此便已夠用，但IPAS卻不能）外，再設一個Eureka（我發現了！）變數，當處理至原命題並得出答案時，即立刻跳出所有遞迴（因爲發生錯誤通常都是在此情況下仍繼續推論時），並設置一個存放遞迴層數的變數，當層數過多（快溢位了）時，便把此次參數儲存並退出，待稍後再處理。

- 「真值函數」之定義與處理（95/02/11）

IPAS早期僅能處理AND、OR兩種真值運算，一直忽略維根斯坦提出的「真值運算圖式表達法」，於11日閱及，逐著手實作；但在實際運作時卻無法如預期中得出結論。

經觀察之後發現IPAS將真值運算亦視爲客體處理，致有此疏失；苦思良久之後，決定另建一陣列儲論「名稱」（如“and”）及「圖式」（如“1000”），語法分析改爲「逐字處理」，將處理後的詞都「堆」在一個字串中，當處理至真值運算時即求出此字串之真值，再與後續部份作真值運算即可；但這也使得IPAS無法處理中文（要怎麼斷字呢？），尚待日後改進。

六、研究討論

1.SAPI-IPAS

我在「目的」中已提過，本研究的目的之一即爲比較「硬式」（SAPI、SIR、FETCH）與「軟式」（IPAS、TUTOR、SOS）程式之優劣，在此必須先界定「硬式」及「軟式」之義。

通常「硬式」程式之實作方式（Prolog？）爲「明確界定＜輸入－處理－輸出＞過程中之處理範圍、演算法，及使用「固定資料架構」（擴充性小）與「預設規則型式」（彈性小）以增進效率與程式運作之正確性（註10）」；

「軟式」（LISP？）程式則允許使用者有較大的發揮空間（有時甚至讓使用者以「試誤」（Try-error）法操作）及彈性，但效率較爲不佳（註11）。

這些在SAPI和IPAS的比較上或許可看見一些端倪。兩者之實作題材均爲「符號邏輯」，但表現則迥然不同（見下表）。

比較項目	發展時間	推論規則	處理範圍	使用語法	使用難易	擴充性	速度
SAPI	一年半	內定	較狹	自然語言	較易操作	較小	較快
IPAS	13天[註1]	自行定義	較廣	較不自然	較難操作	較大	較慢

[註1] 這其實對SAPI不甚公平；我在發展IPAS時『照抄』了SAPI的使用界面，而這是修改了近半年才定型的。

此兩類程式之優點是否可能融合呢？我想是的。將推論規則及處理範疇擴充應非難事；操作界面也可以更便於使用（而非「四個字母加上一千個括號」），至於自然語言…我想在「語意分析」方面可能我還有些該看的理論沒看，暫且擱下；速度方面則需要更佳的演算法及搜尋法－此構想當可繼續研究發展，不失爲一可行之方向。

整體而言，雖然SAPI及IPAS於實作上各有優劣，但IPAS之應用範圍較廣，擴充性較大，且尚有較大之後續發展空間，此後應以之爲發展方向，較能有所進展。

2.未來之發展方向

SAPI及IPAS僅爲本研究的實作雛形，目前亦朝頗多尚待改進之處努力；現列舉如下：

- SAPI目前尚無法接受所有真值運算之連接，必須在架構上作調整（約一個月可完成）
- 擬使SAPI處理假言三段論法之外的規則，且由使用者自行定義（約三個月可完成）
- SAPI之「多國語言處理機制」目前僅完成「德文」、「中文」及「英文」等三種，今後將嘗試處理不同文法的自然文字／語言，及Lojban（邏輯語）等人造語言（約半年）
- IPAS之「句式鑑別」雖較「關鍵字搜尋」爲佳，但仍需靠「字詞數」與「關鍵詞性」等因素判定，以致無法接受中文，將來將朝「深層語法」及「語意樹」的方面發展改進（時間未定）
- IPAS之VB版本（IPAS for WINDOWS）尚可利用VB之先天優勢加以改進之功能，如“句式之形象化”（深層語法？）或“以Drag-Drop之方

式進行「動態連結」”（「具有動感的連結過程」）等等，或將客體等「具像化」，達成「語意網路看得見」，etc…（時間未定）

- 將SAPI及IPAS的「查詢」功能擴充，以使之能繪出「語意網路」或「式圖」等較直觀且利於使用者辨別之型式（約半年）
- 將SAPI和IPAS融合（SAPIPAS？），並在進一步熟習「分析哲學」之後，以不同的角度來實作目前的課題。（？）（！）

七、研究結論

1.本研究之價值

本研究具有下列數項應用價值，以下逐項說明：

A.全方位「智庫」的建立：

「智庫」（Knowledge database）是指在某一主題上具有大量知識與適當推導規則的資料庫；它與傳統資料庫的區別在於，使用者能夠以「語意」檢索，而非以往的「關鍵字搜尋」或「整筆查詢」。

SAPI能夠改寫成如此，只要加入一些新的「導向式規則」（i.e.將一些命題導向至不同的知識範疇）並對不同的句式加權、分類即可（事實上，可以把整篇文章輸入並分析）；而「查詢」機制亦使知識間之關係變得明確；這是一個值得探索的方向。

B.資訊語意分析、推論：

本項及上項均爲「將資訊分析爲句式以辨認語意」，但目的與上項迥然不同；IPAS及SAPI可以在適當的改進後，由一定的「公理」（axioms）自行推論出使用者想要的結論，如同現在已有的功能；但句式與推論機制目前尚未成熟，仍有發展空間。

本項目最有價值之處在於「自動推論機制」的建立，屆時電腦將可以更優良的演算法與（以IPAS、SAPI爲起點的）程式架構，考慮人類遺漏之可能性，幫助吾人作語言之邏輯結構分析與推導，從而達到真正「人工、智慧」的境界。

C.將「分析哲學」的課題在電腦上進行實作：

我對於傳統哲學／哲學家有一定程度的涉獵，但對於維根斯坦以降的「現代哲學」－分析哲學，在本研究開始之前尚一無所知；本研究目的之一亦爲「加深對維根斯坦早期理論（註12）之瞭解」，而至今對此已有初步瞭解與實作經驗，且亦爲一頗新之嘗試。

但現代（英語世界中的）哲學學派對於維根斯坦（及其他分析哲學家

，如弗萊格、羅素）學說亦有進一步之推廣與研究，如劍橋、牛津學派、美國分析哲學及後分析哲學等，作者尚未深入研究，今後盼能朝此方向發展。

本研究可以作爲一個具獨創性的起步，讓我們和電腦用不同的角度和觀點一起分析、理解這個世界，探索各種不同的觀點（更重要的是培養電腦有“自己的觀點”）與理念，也爲今後更新的哲學及資訊世界，共同努力！

八、參考書目

1.A.M.Turing（1952） Artifact Intelligence， I.E.E.E.

2.Marvin Minsk（1986） The Society of Mind， M.I.T. Press.

3.Irving M. Copi（1988） Introduction for Logic, The Macmillan com - pany.

4.Jocelyn Paine（1993） The Logic Programming Tutor, University of Oxford.

5.A.V.Aho & J.D.Ullman（1992）, Foundations of Computer Science, Computer Science.

6.M.L.Schagrin, W.J.Rapaport（1985） LOGIC： a computer approach, McGraw-Hill Inc.

7.趙敦華（1988）維特根斯坦，遠流

8.鄺芷人（1987）命題邏輯與布爾代數，文津

9.Bentrand Russell（1985）西方哲學史，五南

10.Budwig Wittgenstein（1921）邏輯哲學論，唐山

11.賀元，賴明宗，劉燈（1994）世紀末軟體革命，傳徵

12.John Krutch著，孫家麟編譯（1987）有趣的人工智慧程式，松崗

註 1：即First-Order Logic(第一級邏輯)，僅以「傳遞法則」進行推論的程式。

註 2：即Babara，三段論法之第一格中最常用的規則。Babara一詞之母音均爲A，即全稱肯定＋全稱肯定＝＞全稱肯定之意，其餘三格則各爲Celarent、Darii，及Ferio。

註 3：SAPI即Syllogistic Argument Process' Interface之縮寫，爲LPIC之「產品」；LPIC即Logistic Process' Interface Compiler，爲產生邏輯推論界面之程式產生器，因此本次科展以SAPI及IPAS爲重點，故不加以介紹。

註 4：IPAS即Immediately Process Applying Symbols之縮寫，爲一多功能邏

輯符號推理程式，其功能見後。

註 5：因爲兩物間的預設關系爲「有交集」，故窮盡了所有可能性：交集爲0、交集爲A或B、交集同時爲A及B，及其他狀況。

註 6：德文中，ich是“我”的主格（受格爲mich），所以當SAPI檢查到ich在後面時，便將主詞與受詞交換。

註 7：《邏輯哲學論》，5.134

註 8：在IPAS中，基本命題就是“單純符號連貫式的排列”，與《邏輯哲學論》完成一致。

註 9：前方的“*”號表示可有可無，“～”表示「非此字不可」，“？”表示任意詞性，x2表示“兩個單詞所組成的詞”；值得注意的是，IPAS沒有「內定」任何詞性的名稱。

註10：“一個不允許使用者犯錯的程式，便是一個天衣無縫（error-free）的程式”——Autrijus T.

註11：其實，這與我們用的電腦Prolog-brain的成分多於LISP-brain的成分也有不少關係。

註12：維根斯坦晚期對於自己早年的著作有些更正與否定之處，故此處稱「早期哲學理論」。

評　語

1.本作品係在個人電腦上發展一套符號邏輯推理程式，以處理符號間的關係，並且有處理語言敍述之能力，所寫的程式命名爲SAPI及IPAS，這兩個程式事實上還不完整，仍在改進中。

2.這個題目所做的，是相當困難的語言處理問題，目前做到的是三段論法，而語言處理還只是簡單雛型，只能處理很簡單的句型，這種以規則推論的方式很難真正解決語言問題，但以國中生的程度，能初步去探討這個問題，非常難得。

3.作者在電腦上的知識相當豐富，對邏輯推論規則等問題，能作多方涉獵，並去瞭解相關的演算法，程度遠超過同年齡學生，並富有研究精神，很值得獎勵。